Mitigation of Topological Inconsistency Attacks in RPL based Low Power Lossy Networks

Anthéa Mayzaud, Anuj Sehgal, Rémi Badonnel, Isabelle Chrisment, Jürgen Schönwälder

2015; **00**:1–20

Mitigation of Topological Inconsistency Attacks in RPL based Low Power Lossy Networks

Anthéa Mayzaud[1,*†], Anuj Sehgal[2], Rémi Badonnel[1], Isabelle Chrisment[1], and Jürgen Schönwälder[2]

[1] *TELECOM Nancy, Université de Lorraine, LORIA UMR 7503, Inria Nancy-Grand Est, Villers-lès-Nancy, France*
[2] *Computer Science, Jacobs University Bremen, Campus Ring 1, 28759 Bremen, Germany*

SUMMARY

RPL is a routing protocol for low-power and lossy networks. A malicious node can manipulate header options used by RPL to create topological inconsistencies, thereby causing denial of service attacks, reducing channel availability, increased control message overhead, and higher energy consumption at the targeted node and its neighborhood. RPL overcomes these topological inconsistencies via a fixed threshold, upon reaching which all subsequent packets with erroneous header options are ignored. However, this threshold value is arbitrarily chosen and the performance can be improved by taking into account network characteristics. To address this we present a mitigation strategy that allows nodes to dynamically adapt against a topological inconsistency attack based on the current network conditions. Results from our experiments show that our approach outperforms the fixed threshold and mitigates these attacks without significant overhead.

Received . . .

KEY WORDS: 1.6 Network Management / Sensor networks; 4.3 Functional Areas / Security management; 5.3 Management Approaches / Autonomic and self management

1. INTRODUCTION

The Routing Protocol for Low-power Lossy Networks (RPL) [1], designed for constrained devices and networks, is expected to find application in multiple areas of the Internet of Things (IoT). Being suitable for various fields like, Industrial Networks [2], Home and Building Automation [3] and Advanced Metering Infrastructure (AMI) Networks [4], it is evident that RPL will be exposed to multiple different operating scenarios, some of which will expose it to malicious attacks.

To overcome such situations, RPL includes security mechanisms that can be used to ensure integrity and confidentiality of messages, however, important features like key-management are left out by the current standard [5]. Furthermore, cryptographic algorithms are known to occupy the most memory and take many CPU cycles, thereby greatly affecting the performance of constrained devices [6] likely to be used in IoT and WSN applications, for which RPL is suitable. An examination of current RPL implementations across Contiki 2.6, TinyOS 2.1.2, RIOT 2013.08 and

*Correspondence to: Anthéa Mayzaud, TELECOM Nancy, Université de Lorraine, LORIA UMR 7503, Inria Nancy-Grand Est, Villers-lès-Nancy, France.
†Email: anthea.mayzaud@inria.fr

Contract/grant sponsor: This work was partly funded by FLAMINGO, a Network of Excellence project supported by the European Commission under its Seventh Framework Program; contract/grant number: ICT-318488.

SimpleRPL 1.0 for Linux, during the course of this study revealed that secure mode of RPL is a feature that is not implemented. This leaves RPL open to multiple attacks wherein a malicious node can manipulate contents of a packet to adversely affect the network.

One such attack is the Destination Oriented Directed Acyclic Graph (DODAG) inconsistency attack, a type of topological inconsistency attack, which entails a malicious node manipulating the RPL IPv6 header options [7] used to keep track of topological inconsistencies. This can lead to an increase in control overhead, which can impact limited energy reserves of constrained devices and decrease availability of an already constrained channel. RPL makes use of a fixed threshold to counteract these DODAG inconsistency attacks. Once a node receives a certain number of packets with the appropriate RPL IPv6 header options, all such subsequent messages are ignored. The RPL standard proposes a value of 20 for this threshold [1], however, does not provide any reasoning as to why this value is recommended.

A malicious node can also use a DODAG inconsistency attack to modify the IPv6 headers of packets it forwards such that it forces the next-hop node to drop these modified packets. This leads to the creation of a black-hole, which is hard to detect and counteract. Such an attack can lead to serious denial of service attacks, or a malicious node could choose to selectively force nodes to drop certain types of control, management, logging or update packets, thereby effecting the overall stability of the network.

Our contribution is the development of an adaptive threshold (AT) mechanism to mitigate effects of such attacks. This initial AT approach has also been further improved to dynamically account for network characteristics while deriving an appropriate threshold for counteracting DODAG inconsistency attacks. Our experimental results show that both our proposed approaches can lead to improvements over RPL's currently used fixed threshold approach. Furthermore, our approaches are able to counteract the black-hole DODAG inconsistency attack scenario while still outperforming the default RPL mitigation strategy.

The rest of this paper is organized as follows. In Section 2 an overview of relevant related work is provided. This is followed by an overview of the RPL protocol in Section 3. The DODAG inconsistency attacks are presented in Section 4, along with the proposed strategies that can be used to mitigate them in Section 5. An evaluation of the DODAG inconsistency mitigation approaches follows in Section 6, before we draw conclusions in Section 7.

2. RELATED WORK

While the study of RPL security is relatively new, the wireless sensor networks (WSNs) community has investigated security in similar environments. The authors of [8] investigate trust to enhance the security of WSNs and also propose to extend their approach to RPL networks [9]. However, such an approach is not useful if malicious nodes perform DODAG inconsistency attacks, because they can easily remain undetected owing to the unaltered control messages they broadcast.

The IETF RoLL working group identified potential security issues in RPL networks and proposed countermeasures [10]. The identified threats were classified into four categories: (1) authentication, (2) confidentiality, (3) integrity and (4) availability. The DODAG inconsistency attack cannot be easily categorized because even though it appears to be an integrity threat, it could also be a malfunction. This implies that any mitigation mechanism should be carefully designed so as to account for network malfunctions as well. The RoLL working group proposed the data path validation mechanism and a fixed threshold [7] approach to mitigate such attacks. However, a data path validation mechanism usually requires nodes to maintain additional state [11], something that quickly reduces the already scarce computing resources at constrained devices.

Studies on other attacks in RPL networks have also been performed in recent years. The authors of [12] explored methods to detect black-hole attacks in RPL networks. Different works investigated packet modification attacks resulting in topology modification or resource depletion, and proposed prevention methods. The authors of [13] studied loop creation when in situations when an attacker modifies the version number field in RPL control messages. An authentication scheme to prevent malicious nodes from modifying important network control data was proposed in [11]. Defense

techniques against sink-holes in RPL networks were explored in [14]. The authors of [15] presented an attack changing the topology by systematically choosing the worst parent in an RPL network. However, none of these works consider DODAG inconsistency attacks.

We have previously performed an initial study on DODAG inconsistency attacks and the effectiveness of the fixed threshold approach [16]. An adaptive threshold (AT) approach was designed during this study to counteract the DODAG inconsistency attacks. However, even though this AT changes based on the rate at which packets are received, it does not take into account network characteristics. Furthermore, certain basic configuration parameters for the AT approach had to be chosen in a somewhat arbitrary way. In this paper we not only extend the evaluation performed on the AT approach, but also develop a new dynamic threshold (DT) approach that also adapts itself based on network characteristics. This DT approach requires no parameters to be chosen arbitrarily, since all information needed is derived from the network itself.

3. THE RPL PROTOCOL

RPL forms a loop-free tree like topology termed a Destination Oriented Directed Acyclic Graph (DODAG), wherein nodes are organized into a hierarchical structure with a root, children, and descendants. Objective functions are used to optimize the topology based on a set of goals, e.g., energy conservation, reduced hop count or best link-quality. Each network can have multiple DODAGs, each optimized with its own objective function and appropriate topology. RPL can also run multiple instances within a network, which leads to each instance having its own set of DODAGs [1]. A node can be a member of only one DODAG in an instance at any given point in time, but it may join multiple instances concurrently.

Formation and maintenance of the RPL DODAG are carried out using (1) DODAG Information Solicitation (DIS), (2) DODAG Information Object (DIO) and (3) Destination Advertisement Object (DAO) control messages. A new node may join an existing network by broadcasting a DIS message to solicit DIO messages, which carry information about the DODAG such as node ID and objective code point. Alternatively, a node may wait to receive DIO messages, which are periodically broadcast, from its neighbors. The DIO transmission periodicity is determined by trickle timers [17].

Upon receiving a DIO message the node calculates its rank by using the objective code point specified in this message. The rank value of a node corresponds to its position in the graph with respect to the root and must always be greater than its parents' rank in order to guarantee the acyclic nature of the graph. If DIO messages are received from multiple nodes, the sender that results in the best rank is chosen as the parent and informed of the decision. To build downward routes a node must send the DAO message, containing routable prefixes, up the tree [1]. As the message propagates upwards, prefixes are aggregated and downward routes become available to parents.

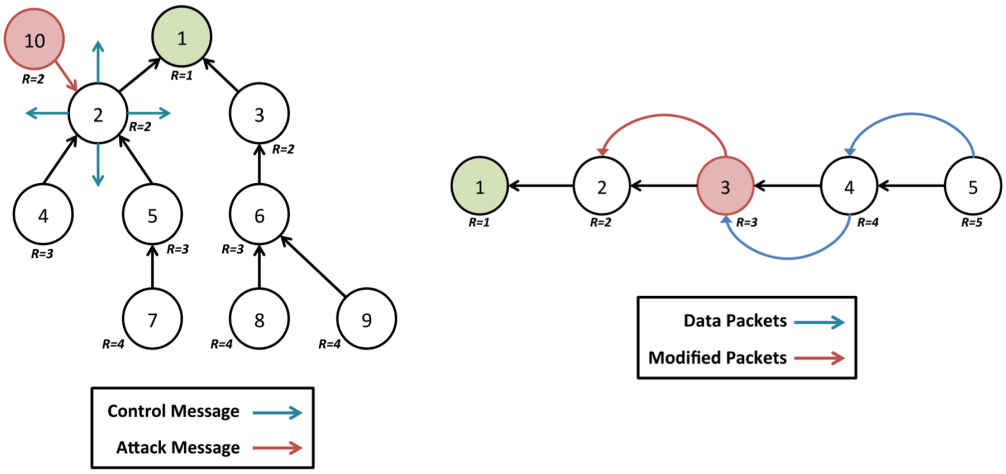
To avoid possible loops, RPL utilizes two basic rules. Any messages traveling downwards in the DODAG, but having originated at a descendant node, are ignored. Also, nodes can normally only change their parents and rank in case of upwards movement. Downwards movement is strictly prohibited, unless it occurs during loop avoidance measures or when a new version of the DODAG is created by the root.

When a loop occurs, RPL provides the *data path validation* mechanism to detect and repair rank related DODAG inconsistencies. This mechanism works by carrying the following flags in the RPL IPv6 header options [7] of multi-hop data packets:

- The *'O' flag* — indicates the expected direction of a packet. When set, the packet is intended for a descendant. Otherwise it is intended for a parent, towards the DODAG root.
- The *'R' flag* — indicates that a rank error was detected by a node forwarding the packet. A mismatch between the direction indicated by the 'O' flag and the rank of sending/forwarding node causes the flag to be set.

A DODAG inconsistency exists if the direction indicated by the 'O' flag does not match the rank relationship of the node from which the packet was received [1]. The 'R' flag is used to repair this problem by setting it, in case it was not set previously, and forwarding the packet. Upon receiving

(2015)
DOI: 10.1002/nem



(a) (b)

Figure 1. DODAG inconsistency attack scenarios. (a) The attacker, node 10, targets node 2 by sending packets with the 'R' flag. This causes node 2 to drop this packet, reset its trickle timer and increase network overhead. (b) The attacker, node 3, modifies packets received from its descendants to contain the 'R' flag before forwarding them. The receiving node 2 drops these packets and resets its trickle timer, leading to increased network overhead and reduced delivery ratio. (R represents node's rank) [16]

a packet with the 'R' flag already set an inconsistency is detected, the packet is discarded and the trickle timer used by RPL is reset [17].

4. TYPES OF DODAG INCONSISTENCY ATTACKS

The RPL data path validation mechanism based on a fixed threshold was designed to improve reliability of the protocol, however, a malicious node can misuse it in order to attack the network; this is called a DODAG inconsistency attack. These attacks can either be used to harm a targeted node, or a malicious node in the routing path may use this approach to manipulated packet headers and cause the next-hop node to drop the modified packet.

4.1. Direct Attack Scenario

A malicious intruder can directly attack its neighborhood by sending packets that have the 'R' flag and the wrong direction set. For instance, if a parent is targeted, the attacker can send packets with the 'O' and 'R' flags set, since packets with 'O' flag are intended for descendant nodes. The parent will detect an inconsistency and thus, drop the packet and restart the trickle timer.

Resetting the trickle timer causes control messages to be sent more frequently which leads to local instability in the network. This increased control message overhead reduces channel availability and increases energy consumption which can lead to a shortened network lifetime in case nodes are battery operated. Since nodes in RPL networks are likely to be resource constrained, they are unlikely to support multi-tasking or large packet buffers. As such, time spent on processing malicious packets could lead to loss of genuine ones.

Figure 1(a) depicts a scenario where such an attack takes place. In this case, a stable network topology of ten nodes is formed using RPL. Node 10 assumes the role of an attacker by sending messages, with the 'O' and 'R' flags set, to node 2, its parent. Node 2 resets its trickle timer, thereby flooding its neighborhood with control messages and affecting nodes 4 and 5 as well.

4.2. Packet Manipulation Scenario

In this scenario, the malicious intruder modifies the IPv6 header of packets it forwards such that the 'R' flag and the 'O' flag representing the wrong direction are set. The receiving node assumes that

Algorithm 1 The default DODAG inconsistency mitigation strategy of a node.

```
if (O = 1 and r_i < r_j) or (O = 0 and r_i > r_j) then
  if R = 1 then
    count_R ++
    drop(M)
    if count_R < λ then
      reset(trickle_timer)
    end if
  end if
end if
```

a DODAG inconsistency has taken place and discards the packet. As a result, the malicious node succeeds in forming a black-hole at the next-hop node. This attack could either be carried out on all packets forwarded by the malicious node, or selectively based on source, destination, or even type of message.

The nodes originating the message cannot easily detect this forced black-hole because the packet is not dropped by their next-hop, but by a node that is at least two hops away. If the malicious node itself were to drop the packets, its children could detect this by enabling promiscuous mode. But promiscuous mode is not an option in such a scenario since in most RPL networks, a node that is two hops away is usually out of radio range as well [12]. This means, that only the attacker is within radio range of the sender and the node that drops the packet, thereby making it nearly impossible for nodes to detect the manipulation. Another problem with the promiscuous mode is that due to the resource constrained nature of the nodes and the lack of multi-tasking or large packet buffers, time spent on inspecting packets leads to loss of packets that should have been handled normally.

In general this approach is a good strategy for the attacker to force another node to drop the packets. Furthermore, if the control packets originating from the malicious node are normal, then the malicious activity is completely hidden. In this scenario, not only does the delivery ratio decrease, but the control overhead of RPL nodes also increases along with deteriorating channel availability and increasing energy consumption.

For example, in the DODAG depicted by Figure 1(b) node 3 is the attacker. Before forwarding data packets from its descendants, nodes 4 and 5, it modifies them such that the 'O' and 'R' flags are set. As a consequence, the node 2 drops them, thereby becoming akin to a black-hole. This causes the delivery ratio for nodes 4 and 5, descendants of node 3 to be severely harmed. Node 2 also resets its trickle timer causing an increase in overhead as well.

5. MITIGATION APPROACH

5.1. Default mitigation

The default DODAG inconsistency attack mitigation strategy of RPL can be seen in Algorithm 1, where i is a node within the graph with a rank of r_i. M represents a packet received by node i from a neighbor j with rank r_j. O and R represent the 'O' and 'R' flags present in M. The variable $count_R$ is the number of received data packets with the 'R' flag set and is initialized to 0. λ is a constant set to 20.

Upon receiving a packet with an inconsistency, the node drops it and resets its own trickle timer. To limit the effects of an attack, the number of trickle timer resets is limited to the recommended constant $\lambda = 20$ [7]. Upon reaching this threshold, malformed packets are dropped but the trickle timer is not reset. $count_R$ is reset every hour, allowing attackers to once again have a higher impact.

This approach limits the impact of a DODAG inconsistency attack, but the value of the threshold $\lambda = 20$ is arbitrarily set. No reasoning is provided to justify this choice or how performance could be improved in case of varying attack scenarios.

(2015)
DOI: 10.1002/nem

Algorithm 2 The adaptive DODAG inconsistency mitigation.

```
if (O = 1 and r_i < r_j) or (O = 0 and r_i > r_j) then
  if R = 1 then
    if count_R < λ(r) then
      count_R ++
      drop(M)
      reset_trickle_timer()
    else if λ(r) ≤ α then
      O ← 0
      R ← 0
      forward(M)
    end if
  end if
end if
```

Algorithm 3 The dynamic DODAG inconsistency mitigation strategy.

```
if (O = 1 and r_i < r_j) or (O = 0 and r_i > r_j) then
  if R = 1 then
    count_R ++
    if count_T < λ(r) then
      if timer_expired(convergence_timer) then
        drop(M)
        start(convergence_timer)
        reset(trickle_timer)
        count_T ++
      end if
    else if r >= 1/ε then
      O ← 0
      R ← 0
      forward(M)
    else
      drop(M)
    end if
  end if
end if
```

5.2. Adaptive Threshold

In order to take into account the current network state and react to varying attack patterns we developed an adaptive threshold (AT) [16], which determines when to stop resetting the trickle timer. Instead of a constant λ, a function $\lambda(r)$ is used, which takes the following form:

$$\lambda(r) = \lfloor \alpha + \beta \cdot e^{-\gamma \cdot r} \rfloor \tag{1}$$

where,

$$r = \frac{count_R}{D_{pkt}}, \quad \alpha = 5, \quad \beta = 15$$

$count_R$ is the number of received data packets with the 'R' flag set and D_{pkt} represents normal forwarded data packets. To allow comparison with the default strategy, the value of β was chosen such that the default $\lambda(r) = 20$. The value α is an asymptote to ensure that threshold never reaches 0. This guarantees that data packet validation is not disabled upon encountering the first packet with an 'R' flag, but only when the situation is deemed an attack.

Since γ impacts the threshold's rate of change, a value is not chosen here. In general, a larger value for γ leads to a smaller threshold being reached quicker. The effect of choosing different values for γ is discussed in Section 6.4.

The adaptive threshold causes $\lambda(r)$ to change based on network conditions. If an attacker is aggressive, the threshold drops quickly and increases slowly once the attacks stop. Unlike with the fixed threshold, $count_R$ is not reset every hour, but rather allowed to increase in the absence of attacks. As such, not only is this approach likely to be better than a fixed threshold within the first hour of an attack, but it should perform significantly better against long running attacks. This also ensures that greater trust is placed in networks where problems have not been encountered for a long time. Of course, a natural limit upon the value of the counter is the bit-length of the variable imposed by the platform. In this case, the counter will reset when the value overflows. If any of the counters overflows, we recommend resetting all counters ($count_R$ and D_{pkt}) so that the algorithm functions as though it was started in a new network.

To counter the packet manipulation DODAG inconsistency attack, an extension was made to the adaptive threshold. Nodes behave normally until the number of messages indicating an

inconsistency becomes greater than the threshold obtained from Equation 1. This situation indicates either an attack against the node, or malfunction of the node forwarding such packets. To rectify the situation, the node clears the 'O' and 'R' flags before forwarding the packets normally. The complete packet manipulation mitigation strategy, combined with adaptive threshold mitigation approach, can be seen in Algorithm 2.

Since no additional resources are used by this approach, the cost of protecting the network against black-hole scenarios is quite low.

5.3. Dynamic Threshold

The adaptive threshold approach relies on a set parameters, which a particular RPL implementation needs to choose. This can lead to sub-optimal optimizations and so we have improved our mitigation approach via the design of a fully dynamic threshold, which is based on network characteristics. The new threshold $\lambda(r)$ used to determine whether the trickle timer should be reset is:

$$\lambda(r) = \lfloor \delta \cdot e^{-\epsilon \cdot r} \rfloor \tag{2}$$

where,

$$r = \frac{count_R}{D_{pkt}}, \quad \delta = 2 \cdot \epsilon, \quad \epsilon = \#neigh$$

As before, $count_R$ is the number of received data packets with the 'R' flag set. D_{pkt} represents normal data packets forwarded by the node.

Normally packets with the 'R' flag set do not arrive at any nodes, because the network is stable and functions as intended. It has been observed, via experiments carried out during this study, that packets with the 'R' flag set arrive only when an attack is performed on the network, or loops form due to serious malfunction of nodes, which is unlikely, unless a software bug exists. Even when the root node initiates a rebuild of the entire network, i.e., a global repair, a maximum of one or two packets containing 'R' flags are received from each child. Any given local neighborhood in an RPL network always returns to stability within two packets containing an 'R' flag, if the problem is a genuine topological inconsistency.

As such, setting δ to twice the number of neighbors (parents and children represented by $\#neigh$ or ϵ in Equation 2) allows the possibility for each link to send up to two packets with an 'R' flag set in case of legitimate loops. $\lambda(r)$ corresponds to the value of δ in a steady state, i.e., when no packets with 'R' flags are received.

Even though not observed during our experiments, it is possible for multiple packets with an 'R' flag to arrive as a result of the same inconsistency. This can be especially true in case a node malfunctions, leading to a loop being formed. Resetting the trickle timer each time a malfunctioning node sends packets with 'R' flags leads to unnecessary overhead, especially since a single trickle timer causes aggressive transmissions of DIOs anyway. To avoid this situation, a $convergence\_timer$ is introduced in this algorithm. This timer is used to ensure that no further trickle timer resets take place within the amount of time it takes for an RPL neighborhood to typically converge. Previous experiments have shown that time for convergence of a DODAG neighborhood increases by about 2 seconds for every additional 10 neighbors [18]. The $convergence\_timer$ is, as such, set to 2 seconds by default but grows based on neighborhood size of a node.

Since the purpose of introducing a $convergence\_timer$ is to block trickle timer resets caused by 'R' flag packets arriving within the time it takes for the neighborhood to converge, it no longer makes sense to compare $count_R$ with $\lambda(r)$ to determine whether a trickle reset must occur. Rather, a new counter that keeps track of the number of trickle timer resets, $count_T$, is introduced. The value of $count_T$ is reset one hour after the first 'R' flag packet is encountered. Instead of λ representing the number of 'R' flag packets allowed before a trickle timer reset occurs, as with the default mitigation approach, it is now the number of trickle timer resets allowed to be caused by 'R' flag packets that arrive while the neighborhood is already considered to be converged. The overall dynamic threshold approach can be seen in Algorithm 3.

(2015)
DOI: 10.1002/nem

The dynamic threshold allows $\lambda(r)$ to change based on network conditions. Like the adaptive threshold approach, this mitigation strategy should perform better against long running attacks. This dynamic threshold approach not only does away with arbitrary constant thresholds, as in the case of the default strategy, but by being based purely upon network characteristics it does away with the need for constant parameters to be chosen before deployment [16] and thereby is more useful in case of unforeseen network conditions as well.

In order to counter the packet manipulation scenario using the dynamic threshold approach, Algorithm 3 allows a node to forward packets with the inappropriate flags in some situations. Firstly, as long as $count_T$ is lesser than $\lambda(r)$, i.e. as long as a direct DODAG inconsistency attack or genuine topological error is being corrected, all packets containing the incorrect flags are dropped. However, once this mitigation is over, it is deemed that the network should have returned to normal and any further attack could be a packet manipulation DODAG inconsistency attack. As such, if more than $1/\epsilon$ traffic received by the node contains the 'R' flag, then this is considered a packet manipulation attack. In this case, having given enough chances for the network to fix itself, the node clears the flags and forwards the message normally. For example, if a node has three neighbors and $\frac{1}{3}$ of its traffic contains 'R' flags, the node considers itself to be the target of a packet manipulation attack. This situation is then resolved after the direct attack mitigation threshold, $count_T$, is exceeded.

6. EXPERIMENTAL EVALUATION

The Contiki 2.6 [19] operating system was chosen in order to perform an evaluation of the DODAG inconsistency attacks since it provides an RPL implementation that works on multiple platforms. The TelosB, also known as the TMote Sky, was used as the development platform of choice since its computational resources allow it to function as an RPL router node with the Contiki RPL implementation. To allow evaluation under multiple scenarios, instead of building a topology of actual nodes, the compiled binary for a TelosB was used in the Cooja [20] simulator provided by Contiki with Unit Disk Graph radio attenuation and scattering model (UDGM). This approach provides a method of testing the adaptive threshold under conditions where the lossy IEEE 802.15.4 channel does not cause packet loss. This allows evaluation of our approach under ideal conditions, with no external characteristics causing bias in the results. Utilizing Cooja is quite close to using real hardware since it uses the MSPSim software to emulate the MSP430 architecture and the performance of a MSP430F1611 microcontroller, which is utilized by the TelosB.

6.1. Simulation validation

Even though the Cooja approach is expected to be close to real performance, a validation of the simulation is important. As such, the topology shown in Figure 1(a) was setup using real TelosB motes, with node 1, the DODAG root, acting as the sink. All other nodes were configured to send messages to the sink every six seconds. An additional per transmission random back-off period of up to six seconds was utilized to avoid packet collisions and add a degree of irregularity to the transmission scenario. The dynamic threshold mitigation mechanism was deployed to all nodes.

The attacker node, i.e., node 10 in Figure 1(a), was setup to periodically send packets with the 'O' and 'R' flags towards the sink. This period was varied from 20 to 90 messages sent per hour. The experiment was repeated five times for each attack frequency and lasted for a duration of one hour each time. The amount of outgoing packet overhead at the attacked node, which is the number of DIS, DIO and DAO messages, for varying number of attacks per hour can be seen in Figure 2. The same experiment was carried out in Cooja as well.

It is clear from the plot that the results provided by Cooja are within the deviation range of the overhead seen in a network of real motes. This indicates that the Cooja simulations provide results which closely mimic reality. Furthermore, the overhead reported by Cooja is on average higher than in reality because the IEEE 802.15.4 channel causes packets to be lost in a deployment of real motes, whereas this does not occur in Cooja.

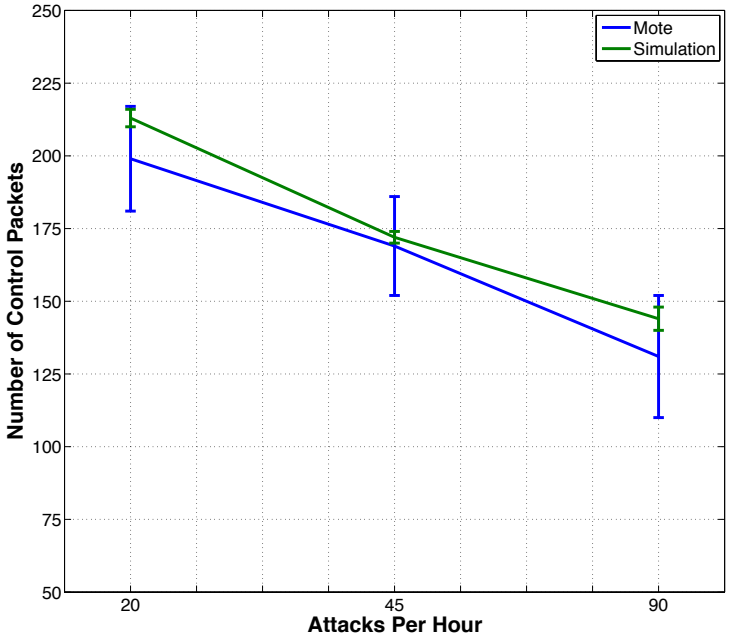


Figure 2. A comparison of the per node outgoing packet overhead (DIS, DIO, DAO) for node 2 in the topology from Fig. 1, in case of a network of real TelosB motes (error-bars for average from 5 runs) and simulated in Cooja.

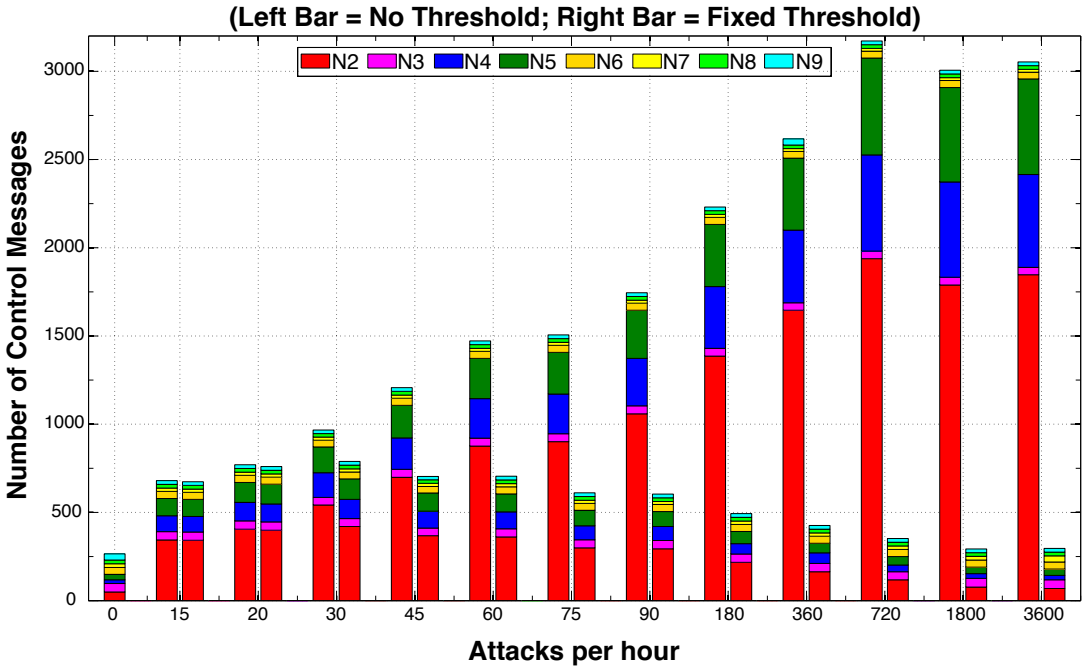


Figure 3. Total control message overhead experienced by a network per node when no mitigation strategy and the default mitigation strategy are used. ($N2 \ldots N9$ represent nodes $2 \ldots 9$ in Fig. 1(a))

(2015)
DOI: 10.1002/nem

A larger topology was not used since the effect of a DODAG inconsistency attack is limited mostly to the targeted node. Its children and further descendants are affected only to a small degree. A larger topology would only make the overhead greater, but not change the patterns observed with this topology.

6.2. Direct attack mitigation

Using the same basic experimental setup as in Section 6.1 the performance of the adaptive and dynamic threshold mitigation approaches were evaluated using simulations. The attack frequency was varied from 15 to 3600 attacks per hour.

It is clear from Figure 3 that the overhead increases due to such attacks. As expected, the more aggressive the attacker, the higher the overall message overhead in the network. Node 2 experiences the largest increase in control messages since it is directly targeted. Nodes 4 and 5 also experience an increase due to being direct descendants of node 2. The control message overhead can increase by over 1100% in worst cases (720 attacks per hour). It can also be seen that at one point the attacker becomes so aggressive that the overhead actually stabilizes, and even reduces do to collisions that occur in the network due to a high number of packets being transmitted. The figure makes it clear that using no mitigation approach leads to the overhead increasing many-fold. As such, it is very important to mitigate DODAG inconsistency attacks.

When using the fixed threshold to mitigate DODAG inconsistency attacks, we can see from Figure 3 that worst case overhead reduces by nearly 85%. Aggressive attacks cause the threshold to be reached faster, causing lower overhead in these scenarios. As such, the best strategy for an attacker is to remain as close to the threshold as possible, as is evident from the 20 attacks/hr scenario. Since the counter for DODAG inconsistencies is reset every hour, by remaining close to the fixed threshold the attacker can do maximum damage and the nodes have no recourse. While a threshold is undoubtedly useful in mitigating such attacks, adapting it to current network conditions would not allow an attacker to keep just below a well-known value and neither would counter resets give the attacker another window of opportunity. The adaptive threshold approach provides such a solution.

From Figure 4, it is clear that the adaptive threshold is more successful in reducing control message overhead than a fixed threshold. An aggressive attack causes the adaptive threshold to reduce rapidly, thereby limiting the impact of the attack. This results in slower attacks being the best strategy. We can also see that 20 attacks/hr is the best strategy for an attacker because the values of α and β were chosen to model the recommended default of 20 in a steady state. However, if the values of these coefficients are changed, so will the periodicity of the optimal attack pattern. For the most aggressive attacks the differences are not so significant since the fixed threshold is quickly reached. The adaptive threshold is between 8% ($\gamma = 20$) to 13% ($\gamma = 25$) better, even in the worst case scenarios.

Figure 5 shows that the dynamic threshold is able to reduce overhead by 20% for aggressive attacks and 50% for slow attacks, when compared to the default fixed threshold approach. Comparing Figures 4 and 5, we can see that the advantage of both approaches is almost the same for aggressive attacks (above 90 attacks per hour). However the dynamic threshold has better results for slower attacks, which means in by using the dynamic threshold approach the attacker cannot use a strategy that overcomes mitigation.

Since the value of $count_R$ is not reset every hour for the adaptive and dynamic thresholds, the attacker does not have a future window of opportunity for causing increased damage. Both these thresholds increase in the absence of an attack, and as such the adaptive and dynamic approaches mitigate long running attacks even better. Results from a two hour long experiment can be seen in Figure 6. Only results from the directly attacked, node 2, are depicted.

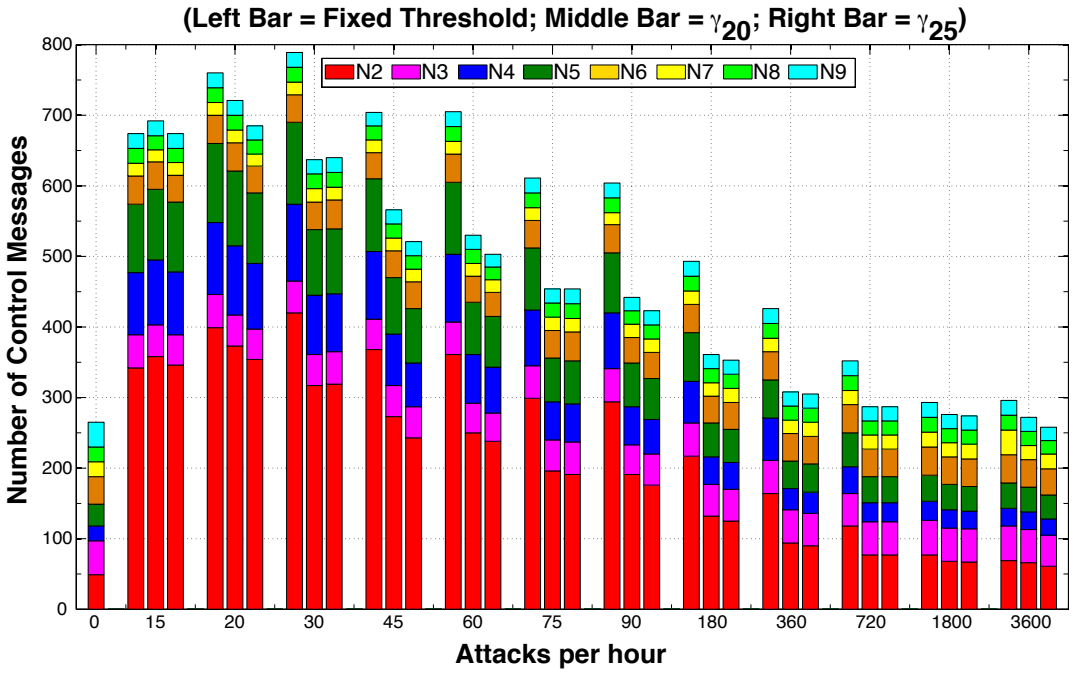


Figure 4. Total control message overhead experienced by a network when the default mitigation strategy and adaptive threshold $\gamma = 20$ and $\gamma = 25$ are used. ($N2 \ldots N9$ represent nodes $2 \ldots 9$ in Fig. 1(a))

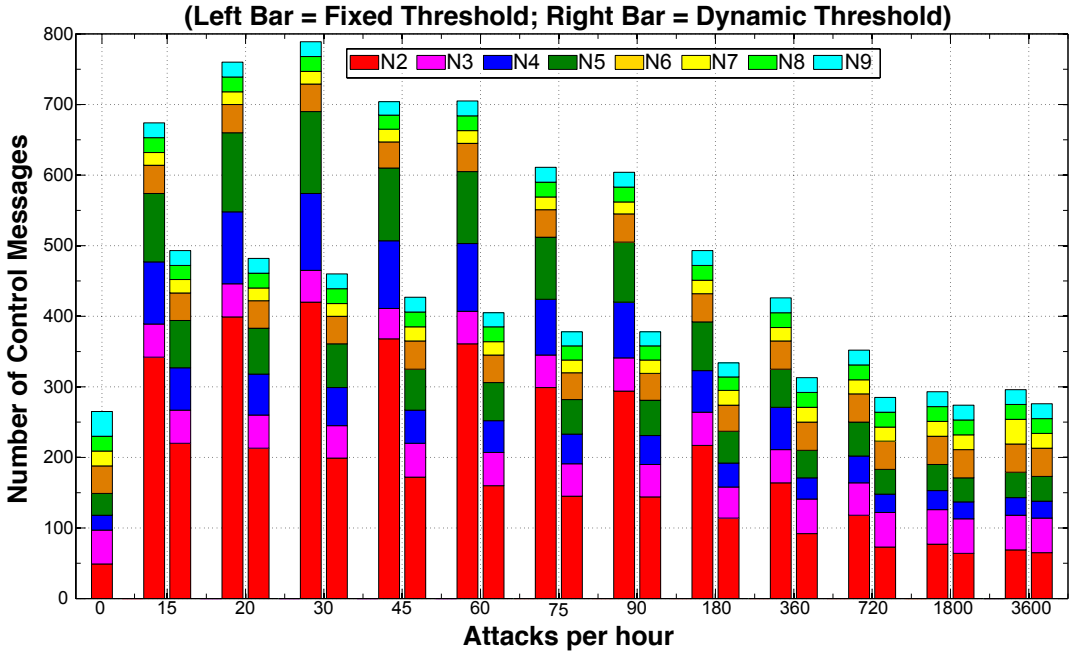
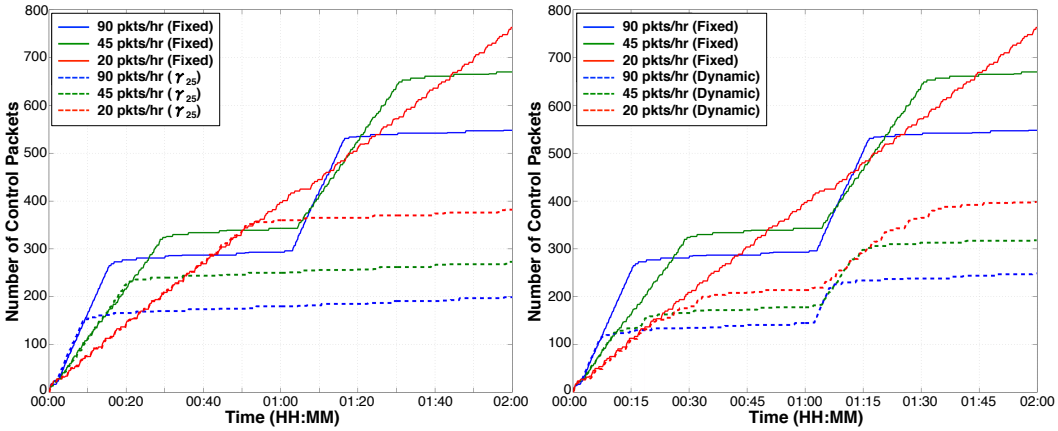


Figure 5. Total control message overhead experienced by a network when the default mitigation strategy and the dynamic threshold are used. ($N2 \ldots N9$ represent nodes $2 \ldots 9$ in Fig. 1(a))

(2015)
DOI: 10.1002/nem



(a) (b)

Figure 6. Time-line of outgoing packet overhead (DIS, DIO, DAO) experienced by the attacked, i.e., node 2 in Fig. 1, when the rate of attack is varied from 20 to 1800 packets per hour. Comparison of (a) fixed threshold and adaptive threshold time-lines ($\gamma = 25$), (b) fixed threshold and dynamic threshold.

In Figure 6 we compare the fixed threshold approach to adaptive and dynamic thresholds approaches. When using a fixed threshold the control messages increase quickly till the threshold is encountered. They then grow at a slow rate, following the trickle timer pattern until the 1 hr mark, when the counter is reset. Once again, the control messages increase quickly until the threshold is encountered. This behavior causes a high control message overhead. The only exception is the period of 20 attacks per hour, because at this rate the threshold is never encountered, thereby causing the largest overhead growth.

On the other hand in Figure 6(a), the limit is reached much faster with the adaptive threshold, due to the exponential growth of the function. Coupled with a non-reseting counter, this leads to between 45%-55% savings in the control message overhead. Those results depend on the value chosen for γ (discussed in Section 6.4). We notice a similar tendency in Figure 6(b) with the dynamic threshold approach. Instead of rising quickly in the second hour, as happens in case of the fixed threshold, overhead increases slowly with the dynamic threshold since the r from Equation 2 increases slowly. The saving for the different attack patterns is around 45%. In comparison with Figure 6(a), we can see that the adaptive threshold has slightly better results. This is due to the γ chosen here and also because $count_T$ in Algorithm 3 is reset every hour to allow legitimate 'R' flag packets from neighbors to be correctly handled.

Unlike with the adaptive threshold, the increase in overhead will continue after the second hour while using the dynamic threshold. This makes it seem like it might be better to use the adaptive threshold, however, this is not necessarily the case. Firstly, the dynamic threshold is able to mitigate packet manipulation attacks, unlike the adaptive threshold algorithm. Furthermore, the adaptive threshold requires setting the γ value, which needs to be learned empirically for every node in the network if optimal performance is desired. The dynamic threshold does not require any such empirically learned values to be configured. As such, because we gain more flexibility and mitigation of an additional type of attack, the dynamic threshold algorithm is recommended over adaptive threshold.

6.3. Packet manipulation mitigation

To evaluate the effect of our mitigation approaches on packet manipulation attacks (Algorithms 2 and 3), the topology shown in Figure 1(b) was setup in Cooja, with node 1, the DODAG root, acting as the sink. All other nodes, except the attacker were configured to send messages to the sink at rates varying from 5 to 20 packets per minute. The packet sending rate is varied, because the attacker, i.e., node 3, silently modifies the option headers of the packets it forwards, rather than originating a direct attack.

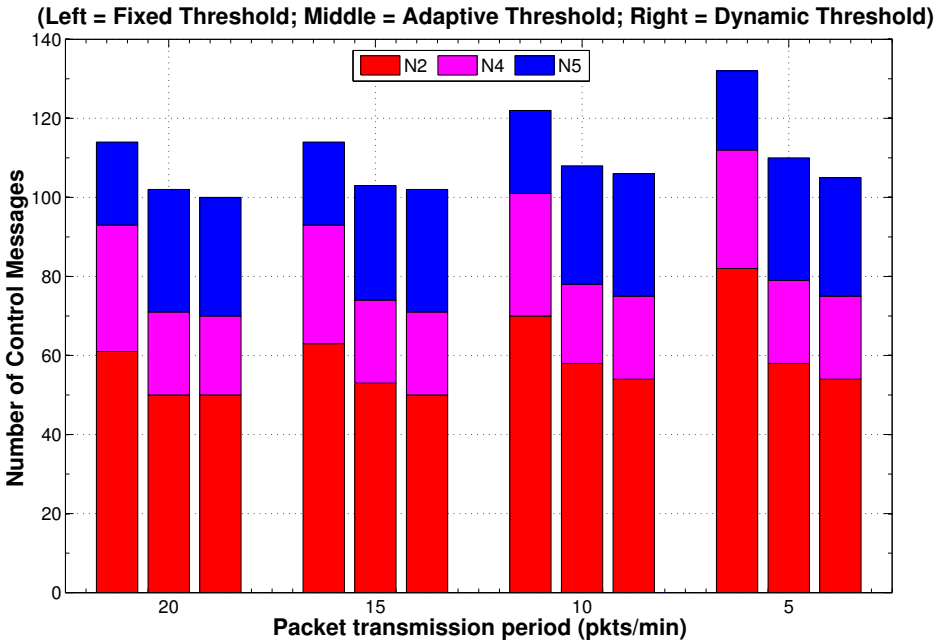


Figure 7. The total control message overhead experienced by a network operating with a fixed threshold, an adaptive threshold ($\gamma = 25$) and a dynamic threshold when a black-hole is formed at node 2 from Fig. 1(b). ($N2 \ldots N5$ represent nodes $2 \ldots 5$ in Fig. 1(b))

The nodes were configured to use the adaptive threshold, then the dynamic threshold for mitigating packet manipulation and the overhead caused by DODAG inconsistency attacks. Results in Figure 7 show that the adaptive and dynamic thresholds reduce overhead in the network. Compared to the default fixed threshold approach a reduction up to 30% can be achieved.

However, the main effect of packet manipulation attacks is not only to increase overhead, but to cause the next-hop node to drop packets of the attacker's descendants. This black-hole created at the next-hop node can severely impact the overall delivery ratio of packets, since none of the packets from the attacker's descendants will reach the sink. Without a black-hole mitigation approach the overall delivery ratio is only about 33%. This is because only packets from node 2 reach the sink, while the attacker forces node 2 to drop all packets sent by nodes 4 and 5.

On the other hand with the adaptive threshold strategy the overall delivery ratio increases to just above 99%, because node 2 no longer drops packets from node 4 and 5 once the threshold is reached. The dynamic threshold approach also has a similar performance, with the delivery ratio being above 99%. However, since the dynamic threshold's working depends upon the size of a node's neighborhood, it is also important to check the effect this can have upon the delivery ratio. Figure 8 shows the delivery ratio for different neighborhood sizes of node 2. The experiment was repeated five times in order to obtain a standard deviation. In case of two neighbors, which corresponds to the simple scenario being tested, we can see that the delivery ratio is above 99% as well (the effect of different neighborhood sizes is discussed in Section 6.4.2). These results speak strongly in favor of mitigating packet manipulation based DODAG inconsistency attacks via an adaptive or dynamic threshold approach.

It is important to note here that in certain situations it is possible that the packet manipulation attack mitigation might take a long time to start up. This is only the case when a network has not been attacked via packet manipulation for an extended period of time. In this scenario, it would take $1/\epsilon$ packets for the mitigation to start, which can be quite a large number if the network has been operating normally for a long time. This situation could be resolved by resetting the D_pkt counter periodically. Ideally, the reset period for this counter should be chosen based upon the packet

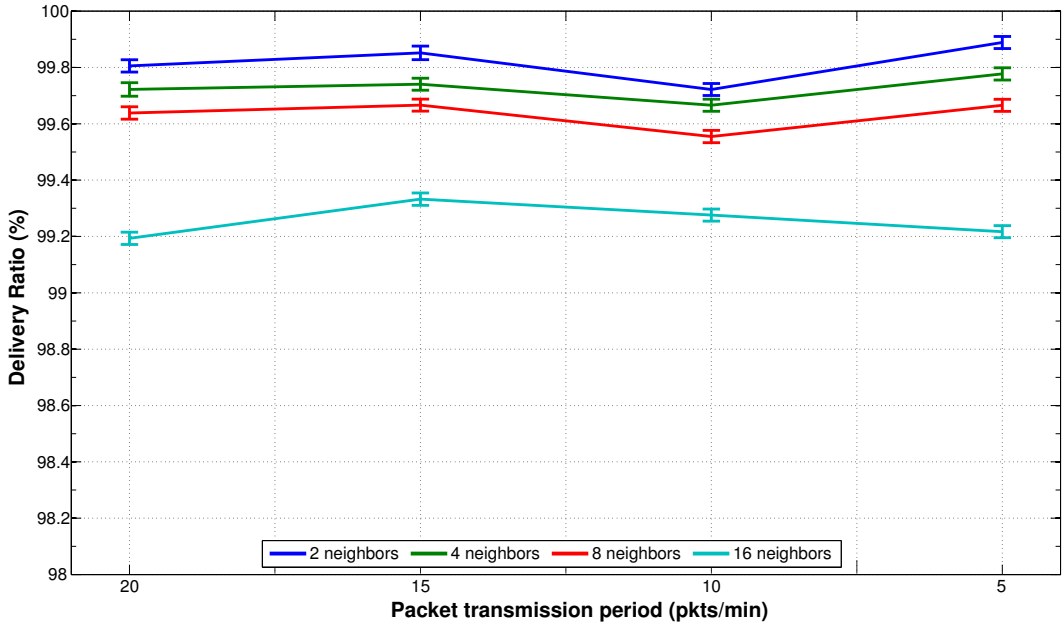
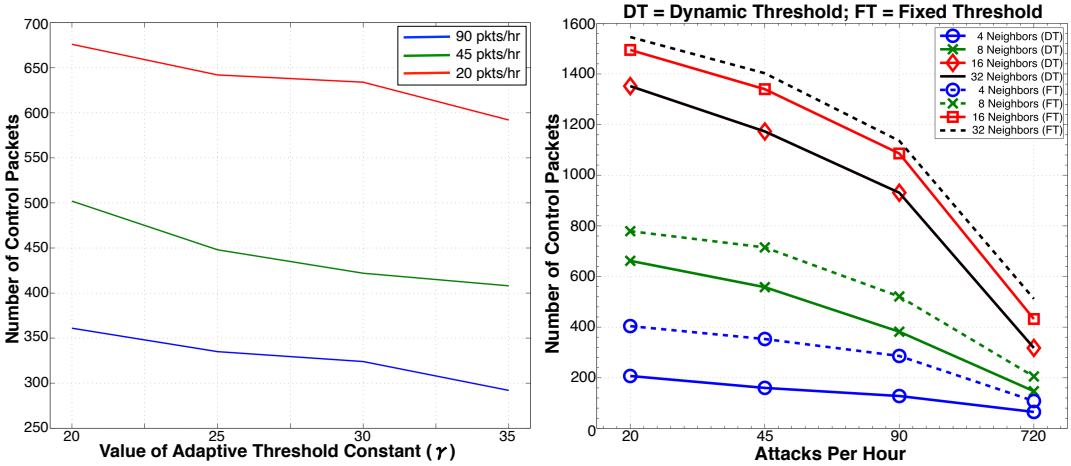


Figure 8. Delivery ratio of the whole network in Fig. 1 (b), when the dynamic approach is use, the rate of data packet transmission is varied from 5 to 20 packets per minute; and neighborhood size of node 2 changes between 2–16 nodes.



(a) (b)

Figure 9. (a) The effect of different values for γ on the total control packet overhead experienced by the attacked node, i.e., node 2 in Fig. 1(a), under multiple attack patterns. (b) Outgoing packet overhead (DIS, DIO, DAO) experienced by the attacked, i.e., node 2 in Fig. 10, when the rate of attack is varied from 20 to 720 packets per hour; and neighborhood size changes between 4–32 nodes.

transmission rate. For more frequent packet transmissions, the value of the reset period should be smaller.

Even though this slow startup may seem like a disadvantage, it should be noted that the dynamic threshold approach will eventually discover the packet manipulation attack and then mitigate it for the future. Furthermore, in most situations, such an attack would be discovered and resolved normally. As such, the dynamic threshold is recommended over the other approaches, since unlike those it will mitigate the packet manipulation attacks and save upon significant overhead once the attack is discovered.

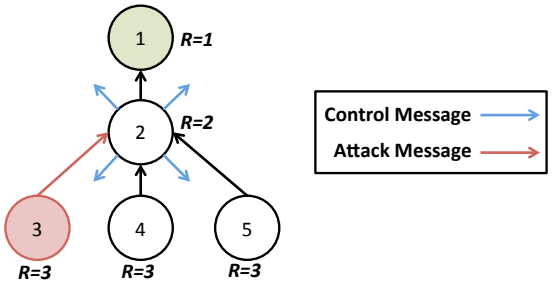


Figure 10. DODAG inconsistency attack scenario used to study the effect of neighborhood size on the dynamic threshold. The attacked node 2 has a neighborhood size of 4 in the shown topology. Similar topologies with neighborhood sizes of 8, 16 and 32 were also evaluated. Nodes 1, 2, and 3 retain their function of root, attacked and attacker, respectively, for all neighborhood sizes. (R represents a node's rank)

6.4. *Effect of Parameter Values*

In the adaptive and dynamic threshold, the calculation of the threshold $\lambda(r)$ depends on different parameters. In this section we discuss the effects of those parameters on the mitigation efficiency.

6.4.1. Adaptive Threshold - Effect of γ Given the same attack periodicity, the value of γ in Equation 1 determines the rate at which the threshold changes. Experiments were run with $20 \leq \gamma \leq 35$ to gain insights into its impact. Values larger than 35 have not been used because larger values of γ result in the threshold dropping too quickly. This leads to situations where even a single packet with the 'R' flag causes the trickle timer resets to stop. This means that genuine malfunctions will no longer be repaired either. In our tests we observed that over values of 35, this situation was encountered frequently. Below 20, the threshold reduces too slowly, thereby making it too permissive and increasing the likelihood of an attack working.

As can be seen in Figure 9(a), by increasing the value of γ the overhead, even in the case of the most efficient attacker, can be further reduced by around 10%. This means that higher values of γ are able to offer more significant savings in the overhead. Our recommendation is to keep the value of γ between 20 and 35 so that the algorithm is neither too permissive nor too aggressive.

As such, the temptation to use a larger value for γ might be high, but it is important to keep in mind that a rapidly reducing threshold might also impact the repair of genuine loop conditions. It would, therefore, be best to remain conservative in choosing a value for γ.

6.4.2. Effect of neighborhood size The performance of the dynamic threshold approach is closely tied to the size of an attacked node's neighborhood. This makes it important to study the effect of varying neighborhood sizes on the dynamic threshold. The same attack and data packet transmission scenarios from Section 6.1 were used with the topology from Figure 10 to evaluate the impact of changing neighborhood sizes. The number of neighbors for node 2, targeted by node 3, was set to 4, 8, 16 and 32 neighbors. A larger neighborhood size was not evaluated since Contiki can only track about 30 neighbors [21]; furthermore, due to the limited resources on the TelosB mote, maintaining a list of large number of neighbors can lead to a node being out of resources.

The overhead experienced by node 2 under different neighborhood sizes and attack patterns can be seen in Figure 9(b). The dynamic threshold outperforms the default fixed threshold approach, in all neighborhood sizes. In fact, the savings to be achieved are between 20-50% and are mostly impacted by the variation of r. The one major advantage the dynamic threshold approach appears to have is that after reaching a neighborhood size of at least 16 nodes, the control overhead does not increase more significantly in case of a larger neighborhood. This is because larger neighborhoods cause the threshold to get smaller faster.

In Figure 9(b) the curves for neighborhood sizes of 16 and 32 nodes, while using the dynamic threshold, are the same because the threshold values obtained in these cases are nearly the same. This happens because of the impact of neighborhood size on the calculation of the threshold. Since

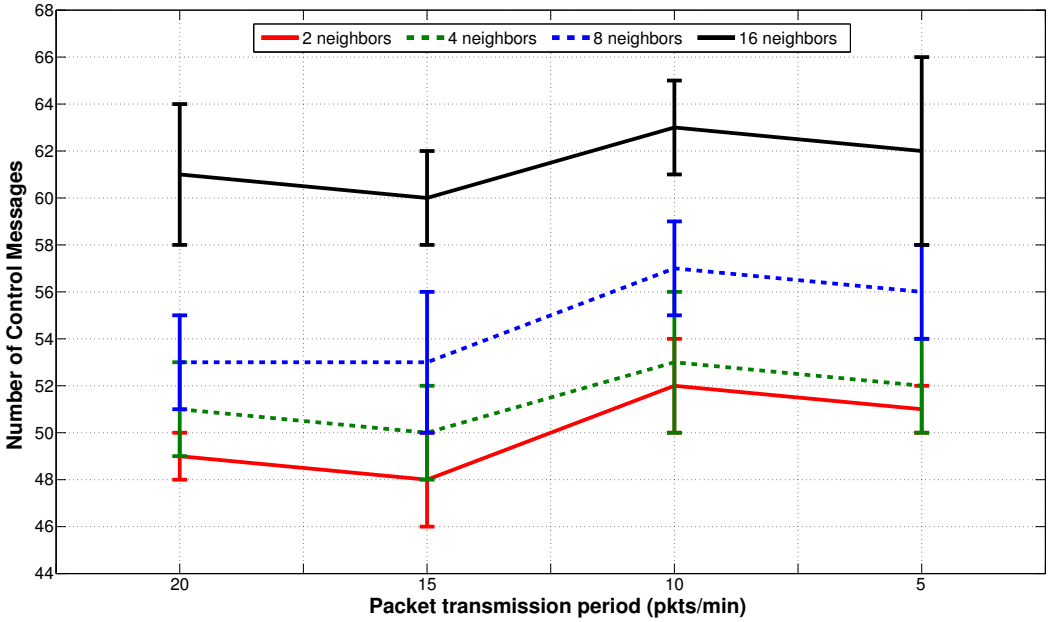


Figure 11. Outgoing packet overhead experienced by node 2 in Fig. 1 (b), when the rate of data packet transmission is varied from 5 to 20 packets per minute; and neighborhood size changes between 2–16 nodes.

larger neighborhood sizes cause the threshold to reduce quickly, in case of 16 and 32 nodes, the threshold reaches its minimum value at the same time. As such, the dynamic threshold leads to lesser overhead in large neighborhood sizes.

The effect of varying number of neighbors was also studied in the packet manipulation scenario. The same simulation scenario as Section 6.3 was used, the number of neighbors for node 2 was set to 2, 4, 8 and 16 neighbors. Larger neighbors were not studied since, as previously mentioned, the impact of larger neighborhoods is not significant. Figure 11 shows the overhead experienced by the node 2 for different packet transmission patterns. The overhead increases according to the number of neighbors, this is because the threshold allows more resets to occur when the size of the neighborhood is larger as designed in the Equation 2. Figure 8 presents the delivery ratio for different packet transmission patterns when the number of neighbors of node 2 is varying from 2 to 16. The delivery ratio decrease when the number of neighbors is increasing in accordance with the Algorithm 3. However even if the size of the neighborhood is 16 the delivery ratio stay above 99%.

6.5. *Resource consumption*

To evaluate the efficiency of a countermeasure designed for constrained environments it is necessary to assess the cost of the solution in comparison with its benefits.

6.5.1. Computational Overhead Since Equations 1 and 2 replace a constant threshold, the complexity of which is $\mathcal{O}(1)$, it is important to also quantify the impact using an exponential function has upon the overall computation costs. Measuring this impact is even more important since these approaches are expected to be used on resource constrained devices with limited computing abilities.

While running the aforementioned experiments, the time taken to calculate the threshold was also obtained. Table I shows the average computation time required to obtain the thresholds for multiple attack patterns.

Calculation of the dynamic threshold appears to add about 25 ms of computational overhead, and 30 ms for the adaptive threshold. This is because the value of the exponential part of the equation in the dynamic approach is lower than in the adaptive approach.

Table I. Average computation time (ms) to calculate adaptive and dynamic thresholds for different attack patterns (20, 45 and 90 attacks/hour) while using a MSP430F1611 microcontroller operating at 1 MHz on the TelosB platform.

Type of threshold	*20 attacks/hr*	*45 attacks/hr*	*90 attacks/hr*
Adaptive threshold (γ=20)	28 ms	31 ms	31 ms
Adaptive threshold (γ=25)	28 ms	30 ms	31 ms
Dynamic threshold	26 ms	25 ms	24 ms

Table II. Energy model for the CC2420 radio and MSP430F1611 microcontroller operating at 1 MHz on the TelosB platform.

Operation	*Current*	*Voltage*	*Part*
Transmit (T_x)	18.8 mA	2.2 V	CC2420 [22]
Receive (R_x)	17.4 mA	2.2 V	CC2420 [22]
Processing	0.33 mA	2.2 V	MSP430F1611 [23]

Using the `msp430-size` tool it was determined that a node using the *fixed threshold* occupies *41.96 kB (87.4%) of Flash* memory and *8.63 kB (86.3%) of statically allocated RAM*. The *adaptive threshold* approach requires *45.61 kB (95%) of Flash* memory and *8.62 kB (86.2%) of statically allocated RAM*. The *dynamic threshold* approach requires *45.73 kB (95%) of Flash* memory and *8.64 kB (86.4%) of statically allocated RAM*. It is important to note that the base Contiki system is also already a part of this. The almost 8% increase in Flash usage, for both approaches, can be reduced by optimization. On the other hand, there is almost no change in the amount of statically allocated RAM required.

The almost 4 kB increase in Flash memory occupancy is due to the usage of a floating point library for calculation of the thresholds. This negative impact can be reduced greatly by using certain optimization, for example, a lookup table with linear interpolation will save not only flash space but also CPU execution time. Results using such optimizations have not been presented here so that the worst case performance of the algorithms can be quantified.

From the measured worst case values, it can be said that the overall impact of both adaptive and dynamic threshold approaches on computational overhead is quite minimal, especially when taking the gains into consideration. Furthermore, it could also be said that even though the dynamic threshold approach uses a little extra memory, the gains in not having to select constants and yet providing good performance make it a good choice.

6.5.2. Energy consumption From the energy model shown in Table II it can be determined that the amount of energy taken up by the adaptive threshold (γ=25) computation is approximately 22.68 μJ, which is the amount of energy required to keep the processor running for the computation time of 31.25 ms. On the other hand, it can be determined that computing the dynamic threshold uses about 18.14 μJ, since the time to compute the threshold is about 25 ms. This means that for attack frequencies of 20, 45 and 90 attacks per hour, the total energy spent over a period of one hour on computing the adaptive threshold is about 0.45 mJ, 1.02 mJ and 2.04 mJ respectively, for the adaptive threshold (γ=25). On the other hand, this is about 0.36 mJ, 0.81 mJ, 1.63 mJ respectively for the dynamic threshold. Figure 12 presents the energy consumed at the attacked node to calculate the adaptive and the dynamic thresholds. The energy consumed increases by a significant amount when the attacker becomes more aggressive. This is because aggressive attacks lead to more threshold calculations, as such, more energy is consumed.

However, looking only at the energy consumed in calculation of the overhead is not a good measure for energy consumption since such attacks also cause additional packet overhead, which leads to additional consumption by the radio. Since the radio tends to be the most energy hungry device on constrained nodes, it is important to factor this into the energy consumption as well.

(2015)
DOI: 10.1002/nem

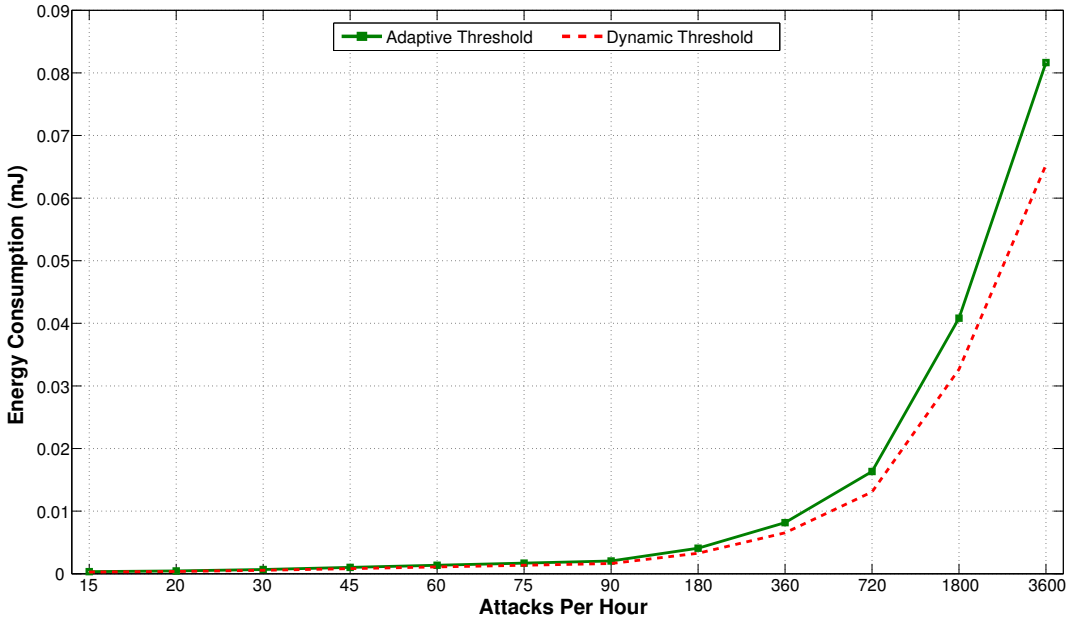


Figure 12. The energy required for adaptive (γ=25) and dynamic threshold to be computed under different attack patterns varying from 15 to 3600 attacks per hour.

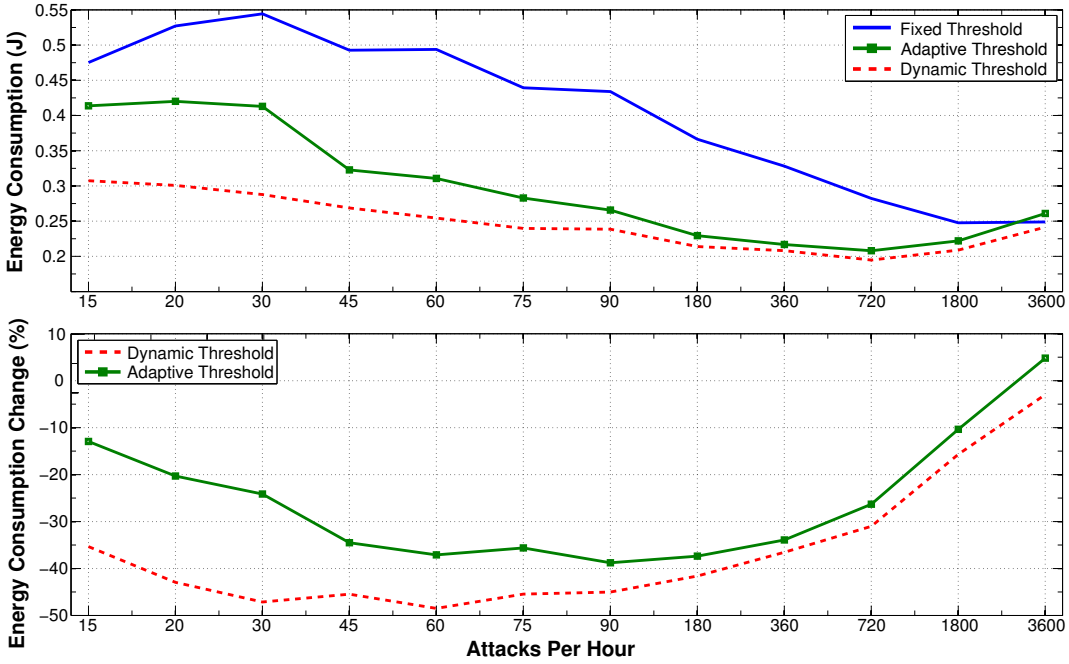


Figure 13. The energy consumption caused by the control message overhead for all nodes in the network and thresholds computation resulting from different attack patterns at the attacked node, i.e., node 2 from Fig. 1(a).(γ=25 for the adaptive threshold)

The upper part of Figure 13 shows the energy consumption caused by the control message overhead and threshold computation for all the nodes in the network. The lower part of Figure 13 presents the change in energy consumption caused by the control message overhead and threshold computation for all the nodes in the network, while the adaptive and dynamic thresholds approaches are used in comparison to the fixed approach. We see that in case of our adaptive and dynamic

thresholds the energy spent by the network to process the control message overhead and the computation of thresholds is less than the energy used for the fixed threshold strategy. However, when the attacker is the most aggressive (3200 attacks per hour) the curves become closer. This is because, in case of aggressive attacks the threshold is computed more often, leading to a higher energy cost. For all attack patterns, the dynamic algorithm has better results than the adaptive threshold as observed in the lower part of Figure 13. In fact, the dynamic threshold approach can provide nearly 50% energy savings in certain attack scenarios.

7. CONCLUSIONS

In this paper we presented two topological inconsistency attacks that are possible in networks that use the RPL routing protocol. It is evident from the experiments we conducted that mitigating such attacks is important to avoid channel congestion and high resource usage. While RPL provides a fixed threshold based approach to mitigate these attacks, the value of the threshold is arbitrary and can be improved by taking into account network characteristics.

Towards this goal we designed an adaptive and dynamic threshold. Both these approaches were evaluated in our study and it was discovered that both outperform the fixed threshold. However, due to the drawback of picking pre-deployment constants that need to be determined empirically for the adaptive approach, the dynamic approach is recommended since it derives all parameters from the network neighborhood size. The performance of our two approaches is quite similar in case of aggressive attacks, however, in all other scenarios the dynamic threshold outperforms the adaptive, thereby making it more suitable for use.

Using our mitigation approach, not only can overhead be reduced between 20%-50%, but even energy savings of up to 50% can be had. In case of black-hole attack scenarios, which are not mitigated by the default RPL approach, our method can improve the delivery ratio to 99% as against 33% for the default RPL mitigation approach. Since these black-hole attacks cannot be mitigated by the adaptive threshold the dynamic threshold is recommended.

REFERENCES

1. Winter T, Thubert P, Brandt A, Hui J, Kelsey R, Levis P, Pister K, Struik R, Vasseur J, Alexander R. RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks. *IETF RFC 6550* March 2012; .
2. Phinney T, Thubert P, Assimiti RA. RPL Applicability in Industrial Networks. *IETF I-D <draft-ietf-roll-rpl-industrial-applicability-02>* October 2013; .
3. Brandt A, Baccelli E, Cragie R, van der Stok P. Applicability Statement: The use of the RPL protocol suite in Home Automation and Building Control. *IETF I-D <draft-ietf-roll-applicability-home-building-06>* December 2014; .
4. Popa D, Gillmore M, Toutain L, Hui J, Ruben R, Monden K. Applicability Statement for the Routing Protocol for Low Power and Lossy Networks (RPL) in AMI Networks. *IETF I-D <draft-ietf-roll-applicability-ami-09>* July 2014; .
5. Seeber S, Sehgal A, Stelte B, Rodosek GD, Schönwälder J. Towards A Trust Computing Architecture for RPL in Cyber Physical Systems. *IFIP/IEEE International Conference on Network and Service Management (CNSM)*, Zürich, Switzerland, 2013.
6. Sehgal A, Perelman V, Kuryla S, Schönwälder J. Management of Resource Constrained Devices in the Internet of Things. *IEEE Communications Magazine* 2012; **50**(12):144–149.
7. Hui J, Vasseur J. The Routing Protocol for Low-Power and Lossy Networks (RPL) Option for Carrying RPL Information in Data-Plane Datagrams. IETF RFC 6553 Mar 2012.
8. Leligou HC, Trakadas P, Maniatis S, Karkazis P, Zahariadis T. Combining Trust with Location Information for Routing in Wireless Sensor Networks. *Wireless Communications and Mobile Computing* 2012; **12**(12):1091–1103.
9. Karkazis P, Trakadas P, Zahariadis T, Hatziefremidis A, Leligou H. RPL Modeling in J-Sim Platform. *Ninth International Conference on Networked Sensing Systems (INSS)*, Antwerp, Belgium, 2012; 1–2.
10. Tsao T, Alexander R, Dohler M, Daza V, Lozano A, Richardson M. A Security Threat Analysis for Routing Protocol for Low-power and Lossy Networks (RPL). *IETF I-D <draft-ietf-roll-security-threats-06>* December 2013; .
11. Dvir A, Holczer T, Buttyan L. VeRA - Version Number and Rank Authentication in RPL. *8th IEEE International Conference on Mobile Adhoc and Sensor Systems (MASS)*, Hangzhou, China, 2011; 709–714.
12. Chugh K, Aboubaker L, Loo J. Case Study of a Black Hole Attack on LoWPAN-RPL. *Proc. of the Sixth International Conference on Emerging Security Information, Systems and Technologies (SECURWARE)*, Rome, Italy, 2012.
13. Mayzaud A, Sehgal A, Badonnel R, Chrisment I, Schönwälder J. A Study of RPL DODAG Version Attacks. *Proc. of AIMS conference*, 2014.

14. Weekly K, Pister K. Evaluating Sinkhole Defense Techniques in RPL Networks. *20th IEEE International Conference on Network Protocols (ICNP)*, Austin, TX, USA, 2012; 1–6.
15. Le A, Loo J, Lasebae A, Vinel A, Chen Y, Chai M. The Impact of Rank Attack on Network Topology of Routing Protocol for Low-Power and Lossy Networks. *IEEE Sensors Journal* 2013; **13**(10):3685–3692.
16. Sehgal A, Mayzaud A, Badonnel R, Chrisment I, Schönwälder J. Addressing DODAG Inconsistency Attacks in RPL Networks. *Proc. of GIIS conference*, 2014.
17. Levis PA, Patel N, Culler D, Shenker S. Trickle: A Self Regulating Algorithm for Code Propagation and Maintenance in Wireless Sensor Networks. *1st Symposium on Networked Systems Design and Implementation (NSDI)*, San Francisco, CA, USA, 2004.
18. Clausen T, Herberg U. A Comparative Performance Study of the Routing Protocols LOAD and RPL with Bi-Directional Traffic in Low-power and Lossy Networks (LLN) . Master's Thesis, Ecole Polytechnique, Centre de recherche INRIA Saclay, Orsay, France 2011.
19. Dunkels A, Gronvall B, Voigt T. Contiki - a Lightweight and Flexible Operating System for Tiny Networked Sensors. *29th Annual IEEE International Conference on Local Computer Networks (LCN)*, Tampa, FL, USA, 2004; 455–462.
20. Osterlind F, Dunkels A, Eriksson J, Finne N, Voigt T. Cross-Level Sensor Network Simulation with COOJA. *31st IEEE Conference on Local Computer Networks (LCN)*, Tampa, FL, USA, 2006; 641–648.
21. Dawans S, Duquennoy S, Bonaventure O. On Link Estimation in Dense RPL Deployments. *7th IEEE International Workshop on Practical Issues in Building Sensor Network Applications (SenseApp)*, Clearwater, FL, 2012.
22. Chipcon AS. CC2420 2.4 GHz IEEE 802.15.4/ZigBee-ready RF Transceiver. *Oslo, Norway* 2004; .
23. Texas Instruments. MSP430F1611 Mixed Signal Controler Datasheet 2006.

AUTHORS' BIOGRAPHIES

Anthéa Mayzaud is a researcher at the MADYNES Team at Inria, France and a second year PhD student at the University of Lorraine, France. She received her M. D. in Computer Science at TELECOM Nancy, France. Her main research interests are security issues in the Internet of Things mostly based on the RPL protocol and lightweight security solutions based on risk management.

Anuj Sehgal is a PhD student of computer science at Jacobs University Bremen, Germany, from where he received an M.Sc. in computer science. His research is currently focused in the areas of Internet of Things and resource constrained networks. His research interests include wireless sensor networks, disruption tolerant networks, underwater acoustic communications, underwater robotics, embedded systems and computer vision. After his undergraduate education at Brigham Young University-Hawaii, he worked as a Systems Engineer at The I.T. Pros in San Diego, CA.

Rémi Badonnel is an Associate Professor at TELECOM Nancy and a research staff member of the MADYNES Research Team at Inria. Previously he worked on change management methods and algorithms at IBM T.J. Watson in USA and on autonomous systems at the University College of Oslo in Norway. His research interests include network and service management, dynamic and autonomic environments, security and defense techniques.

Isabelle Chrisment is a Professor at TELECOM Nancy, University of Lorraine. She received her PhD in Computer Science in 1996 from University of Nice-Sophia Antipolis and her Habilitation Degree in 2005 from University of Lorraine. Vice-head of the MADYNES Reseach team at Inria, her work focuses on network monitoring and security within dynamic networks. Since 2012, she has served as a vice-chair of IFIP Task Force 6.5 related to secure networking.

Jürgen Schönwälder is a Professor of Computer Science at Jacobs University Bremen, Germany where he is leading the Computer Networks and Distributed Systems (CNDS) research group. His research interests include network management, distributed systems, network measurements, embedded networked systems, and network security. Jürgen Schönwälder is an active member of the Internet Engineering Task Force (IETF) where he has edited more than 30 network management related specifications and standards. He serves on the editorial boards of the Springer Journal of Network and Systems Management and the Wiley International Journal of Network Management. He is co-editor of the Network and Service Management series in the IEEE Communications Magazine.